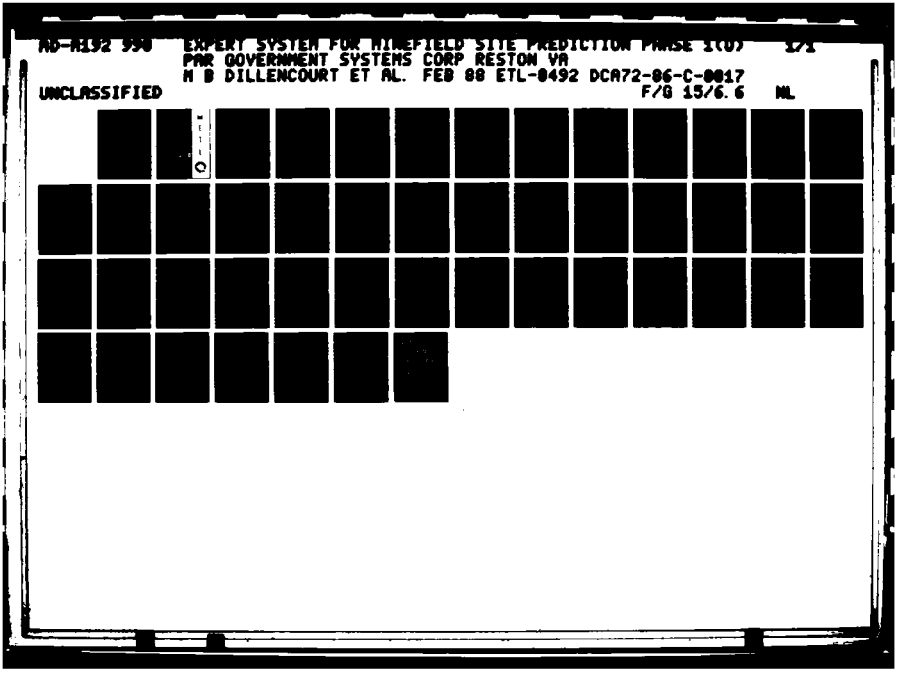
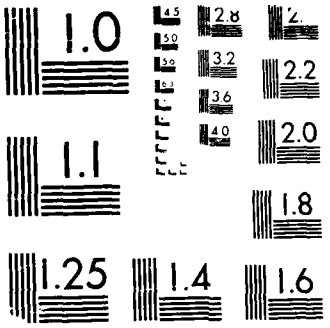

AD-A192 990
EXPERT SYSTEM FOR MINEFIELD SITE PREDICTION PHASE I(U)
PAR GOVERNMENT SYSTEMS CORP RESTON VA
M B DILLENCOURT ET AL. FEB 88 ETL-0492 DCA72-86-C-0017
UNCLASSIFIED
F/G 15/6.6
NL



MICROCOPY RESOLUTION TEST CHART

(4)

ETL-0492

Expert system for minefield site prediction (Phase I)

AD-A192 990

Michael Dillencourt
Jonathan W. Doughty
Anne L. Downs

PAR Government Systems Corporation
1840 Michael Faraday Drive
Reston, Virginia 22090

February 1988

Prepared for:
U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060-5546

88

E T L

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution is unlimited.

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)
ETL-0492

6a. NAME OF PERFORMING ORGANIZATION
PAR Government Systems Corporation

6b. OFFICE SYMBOL (If applicable)

7a. NAME OF MONITORING ORGANIZATION
U.S. Army Engineer Topographic Laboratories

6c. ADDRESS (City, State, and ZIP Code)
1840 Michael Faraday Drive
Reston, Virginia 22090

7b. ADDRESS (City, State, and ZIP Code)
Fort Belvoir, VA 22060-5546

8a. NAME OF FUNDING / SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
DACA72-86-C-0017

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
EXPERT SYSTEM FOR MINEFIELD SITE PREDICTION FIRST YEAR REPORT

12. PERSONAL AUTHOR(S)
Dillencourt, Michael B.; Doughty, Jonathan W.; and Downs, Anne L.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Annual Technical	FROM 2/87 TO 1/88	February 1988	41

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Expert System, Minefield Site Prediction; Quadtree, Terrain Analysis.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The software design of the prototype Minefield Site Prediction Expert Systems (MSPES) is described. The ultimate goal of the system is to emulate the role of a terrain analyst in predicting likely mine sites. The major components of the system are the inference system, the geographic information system, and the user interface. The inference system is driven by a goal-directed backward chaining mechanism. The geographic information system is based on quadtrees. The user interface is menu-driven, and is based on an object-oriented graphics package. This report describes the implementation of the prototype system. It also contains recommendations for the operational system, based on an evaluation of the prototype system. Descriptions of data format conversion capabilities, a detailed description of the geographic processing algorithms, and a complete listing of the rulebase are included as appendices. Keywords:

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
John Benton

22b. TELEPHONE (Include Area Code)
202-355-2723

22c. OFFICE SYMBOL
ETL-RI-I

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Table of Contents

List of Figures

1 Introduction

This report describes work performed during the period from February, 1987 through January, 1988, under Task 2 of Phase 1. It contains a description of the major components of the prototype Minefield Site Prediction Expert System (MSPES).

1.1 Scope of Report

A high-level description of the software architecture has been presented in an earlier document [Barth *et al*, 1987]. This report describes the major system components, and the interfaces and relationships among them. It also addresses the conclusions drawn from building the prototype system and from observing it.

The organization of this report is as follows. Section 2 provides an overview of the system. A description of the various components is presented in Section 3. Section 4 contains some recommendations based on evaluation of the prototype system.

The report is supplemented by several appendices. Appendix A describes the software available for converting data in Standard Linear Format (SLF) to the format appropriate for the QUILT[1] software. Appendix B describes the enhancements to QUILT that were made to support the prototype MSPES. Appendix C contains the source code for the prototype rule base.

This report is not intended to be a User's Manual. Rather, it is intended to provide an overview of the design of the prototype system. It is hoped that this report will prove useful as a basis for evaluating the prototype system, and as a vehicle for recording the lessons learned from building and exercising the prototype system.

---

1 The **QUILT** software package is copyrighted ©1985 by the University of Maryland.

1.2 Scope of Task

Task 2 of Phase I consisted of building a prototype Minefield Site Prediction Expert System, incorporating a knowledge base, and running on a Sun 3 Workstation. The prototype system incorporates rudimentary minefield doctrine by attempting to automate the function of a terrain analyst. A key requirement is that the justification facility be able to explain its conclusions, and that the partial knowledge base be adequate to permit testing of the expert system shell.

For purposes of the prototype, the minefield prediction model used is a simple model derived from concepts in [Falls, 1983]. Based on cross-country movement (CCM) data, areas in which movement through areas of high mobility is restricted to a narrow channel ("canalized areas") and areas along roads are deemed likely mine sites. Areas meeting both these criteria are flagged as very likely mine sites.

The basic ingredients of the prototype system are an Expert System Shell (ERS), the user interface, the graphics packages (SunView and PixRect), and a GIS (QUILT). The capability of ingesting terrain data in SLF format and converting it to QUILT format is also provided. The statement of work originally called for data in MOSS format, but it was agreed that SLF was a more appropriate choice, because this is the accepted DMA data exchange format.

Because the primary goal of the prototype effort was to establish the feasibility of using expert system technology to predict minefield sites, the emphasis was on analysis of CCM data rather than the automation of the map overlay process used to create CCM data. The initial plan was to obtain CCM data and to use that as input. Because of certain difficulties that arose in obtaining CCM data, an alternative approach was used: a Surface Configuration (SLOPE) Overlay was used instead of CCM data. This permitted the prototype to be tested on somewhat realistic data.

1.3 Summary of Work Performed

The major results of the work performed on Task 2 of Phase I were the following.

- *Refined system architecture* . The system is driven by the User Interface and the Inference Engine (ERS). The queries that provide the information necessary for

these driving components (e.g. the geographic queries to QUILT) are implemented as separate processes.

- *A rudimentary rule base.* The rule base that was developed was intended to establish feasibility and is not intended to be complete. A key innovation is the concept of "canalization," which encompasses many of the concepts involved in mine site prediction. The rule base was initially tested manually against simple data to test for gross oversights.

- *Conversion routines.* Routines are provided to translate SLF data into quadtree (QUILT) format. Both area quadtrees and PM quadtrees (which store linear data) are supported.

- *A user interface.* A user interface has been developed and integrated with the expert system inferencing mechanism.

- *Enhancements to QUILT.* Additional primitives have been developed to enhance the basic functionality of the QUILT software. These include processes to report the CCM-class (value), size, and boundaries of the leaf node containing a given point; to determine whether a point lies within a certain threshold distance of a road; to determine whether a node is "canalized" (the concept of canalization is discussed in detail later in this report); and to selectively update portions of a quadtree.

2. Overview of the System

MSPES consists of a number of cooperating processes and data files. The basic components, and the relationships among them, are illustrated in Figure 2.1. MSPTOOL provides the top-level logic that drives the system, the user interface, and the graphics interface. ERS provides the reasoning component of the system. Several processes exist to provide MSPTOOL and ERS with geographic information from the underlying geographic system. The major system data files include help files, rulebases, and the various quadtree files containing overlay and manuscript data.

The individual system components are described in detail in Section 3. In this section, a scenario is presented that describes how manuscripts are created. This scenario illustrates the interactions among the system components. The reasons for implementing the prototype as a collection of communicating processes (rather than as a single process) are also discussed in some detail.

2.1 An Illustrative Scenario

The flow of information among processes during manuscript creation is shown in Figure 2.2. The intraprocess flow (i.e. the data that is passed and returned through subroutine calls) is not shown, but is discussed below.

When the user fires up the **Create Manuscript** application within the **MSPTOOL** process, the application waits for the user to specify an Area of Interest, the name of the rule base to use, and the name of the manuscript to create. The user can specify these parameters in any order, request HELP, or QUIT. When the user "clicks" on the RUN button, **Create Manuscript** checks that all required fields are filled, validates the overwrite of an existing manuscript (if appropriate), checks that all required overlays are available, etc.

Create Manuscript makes a new manuscript directory, copies the CCM file there, creates several files that contain descriptions of what went into the manuscript, and starts up a process called **qdmpupd**. This process is passed the name of the manuscript (quadtree) file. **Create Manuscript** also causes ERS to be started up by calling an Explanation Interface routine, **start_ers**.

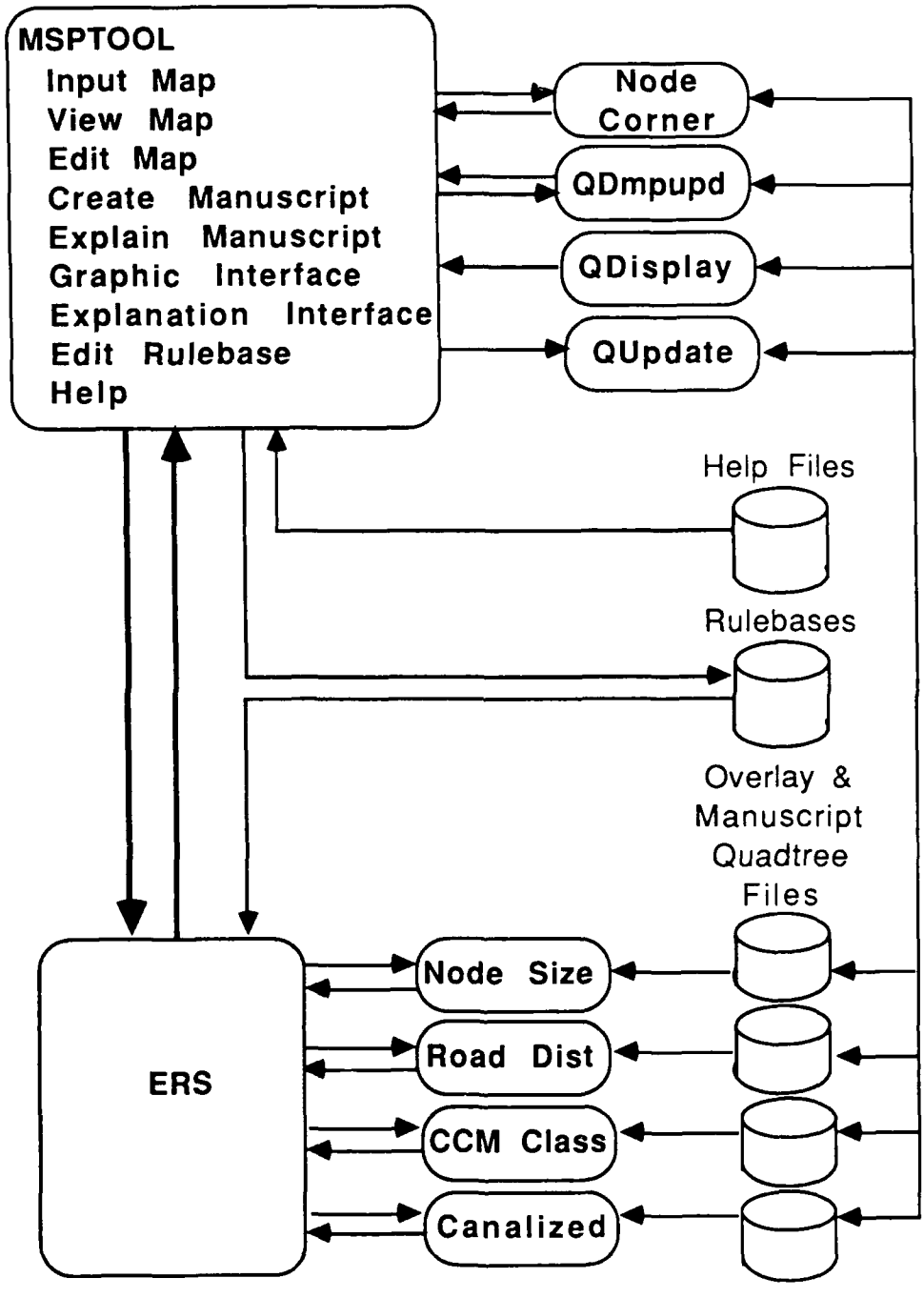


Figure 2.1 The major components of the MINES software.

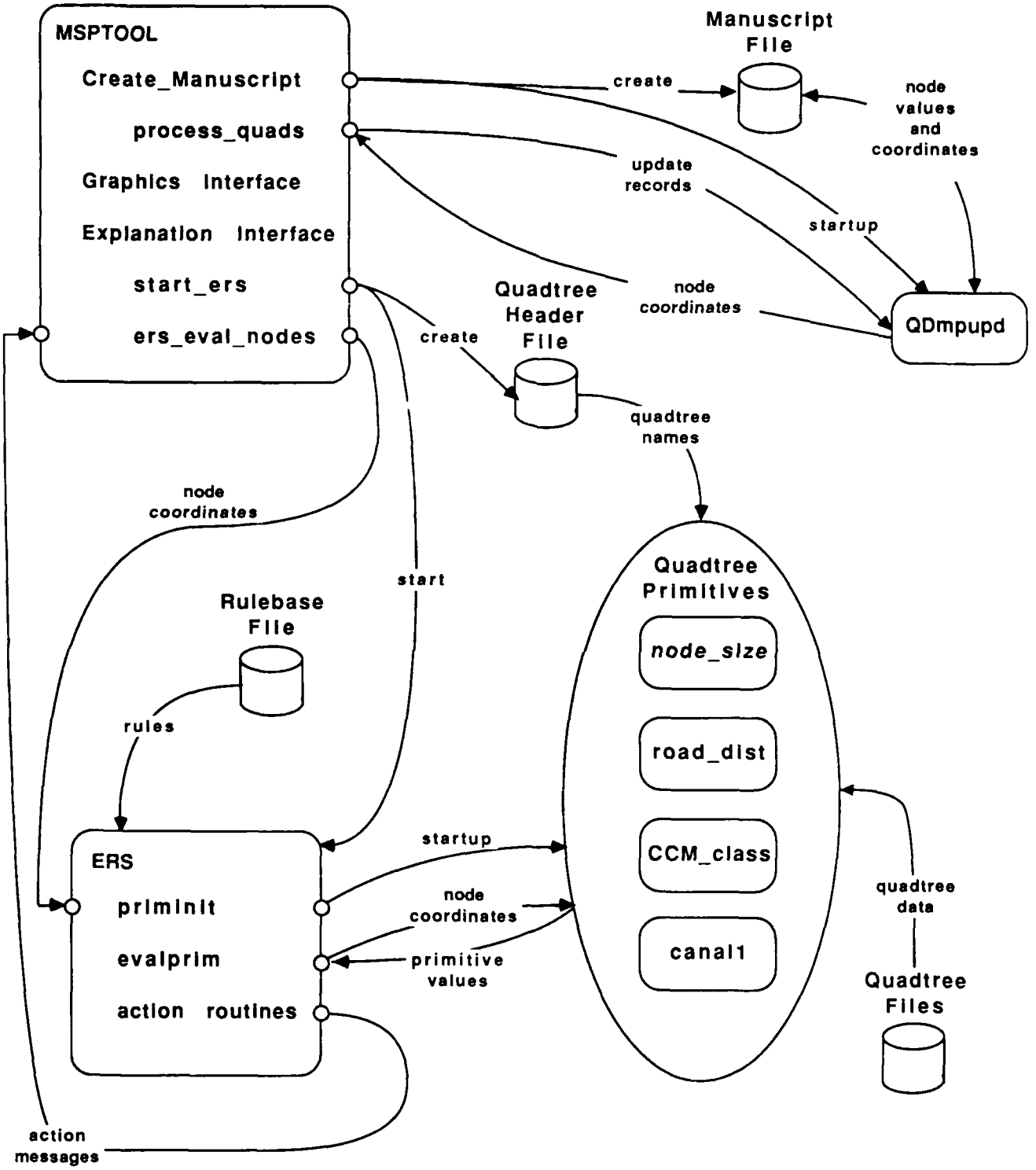


Figure 2.2 Interaction of processes during manuscript creation

Start_ers creates a quadtree header file in the manuscript directory. (The quadtree header file will someday tell the ERS primitive manager which primitives to start up based on what overlays the user has elected to use. For now, we assume that both the CCM and LOC files will be used.) **Start_ers** then starts up the ERS process, and passes to the new process the name of the newly created quadtree header file, name of the rulebase file to use, and the locations of other control files.

When the **qdmpupd** process first gets control, it opens the manuscript file and starts looking for a non-WHITE quadtree node. It causes each such node to be processed by the rest of the system by writing the node coordinates and the size of the node to its standard output. It then waits until it reads on its standard input an update record for that node. The update record specifies a new value, corresponding to the mine site prediction class for the node. **Qdmpupd** then updates the value for the node, if it has changed, in the manuscript file. To facilitate subsequent comparison of the manuscript file with the original CCM file, **qdmpupd** makes sure that the manuscript quadtree has the same "shape" as the CCM quadtree (it does not collapse leaf nodes). Each non-WHITE node is processed in this manner.

Meanwhile, after causing the **ERS** process to be started, **Create Manuscript** has set itself up so that the routine **process_quads** will be called whenever the **qdmpupd** process has output for it. **Process_quads** reads this output, the node coordinates and size, and passes this information along to an Explanation Interface routine, **ers_eval_node,** which writes the current node coordinates to ERS's standard input.

When the ERS process first starts, it opens the rulebase file, along with other control files it was told about on the command line, and initializes its inference net. When it is ready to start evaluating, it calls a routine called **priminit**. **Priminit** waits, reading ERS's standard input for a node coordinate. **Priminit** saves the coordinate value where other primitive management routines within ERS can gain access to it. The first time **priminit** is called it causes the quadtree primitives to be started. When it completes its actions, **priminit** returns control to the body of ERS.

As part of the inference operations specified by the rulebase, ERS calls the routine **evalprim**, identifying the primitive that needs to be called. The node coordinate saved earlier (by **priminit**) is passed to the appropriate primitive. **Evalprim** then gets the result back from the primitive, interprets it, and passes the interpreted information back to ERS.

When ERS decides that certain evaluation criteria have been met, it calls corresponding action routines, which write appropriate messages, such as "Mine likely," to ERS's standard output.

Back in the Explanation Interface of the **MSPTOOL** process, **ers_eval_node** reads the action routine messages written by **ERS**, and filters out "noise" characters such as linefeeds and prompt characters. The word "Mine" delimits the beginning of an action routine recognition sequence. When **Ers_eval_node** detects such a sequence, it calls another routine, **assign_class**, which looks at the remainder of the message, interprets it as a mine site prediction class, and returns the highest class encountered so far for the current node to the calling routine. (This additional step of interpreting several action messages per node and taking the maximum is required because ERS may report that a node is a "possible" mine site before it classifies it as "likely.") Eventually, **ers_eval_node** encounters the ERS prompt character sequence, which indicates that ERS has completely evaluated that node. When this happens, **ers_eval_node** returns the class value for the current node to the caller, **process_quads**.

Process_quads uses the node coordinates and the class value returned by **ers_eval_node** to have the Graphic Interface update the graphic display on the user's screen. **Process_quads** then formats a message to be returned to the **qdmpupd** process, telling it the new value for the quadtree node. **Qdmpupd** then uses this value to update the manuscript file, as described above. Every non-WHITE quadtree node in the manuscript (CCM) file is processed in this fashion.

2.2 Rationale for Separate Processes

It was decided early in the design that distinct functional components would be implemented as separate processes rather than as subroutines of a single, more complex process. This approach is consistent with the Unix design philosophy of communicating tasks with well-defined functions. The approach has the following advantages:

- The individual execution modules are reduced in size, which tends to decrease system paging overhead and to minimize the danger of running into process size limitations.

- The danger of the subsystems colliding in their use of global system resources (e.g., files, external variables with the same name) is reduced. This was a particularly important consideration for the prototype MSPES, since a major portion of the effort consisted of integrating existing software (e.g. QUILT and ERS), rather than developing it "from scratch." Isolating software components by making them separated processes reduces the amount of effort that must be spent verifying that they do not use incompatible programming conventions.

- The interface between subsystems is limited to explicit messages. There is no communication through shared memory. Moreover, it is not possible for an error in one process to destroy the address space of another process. Consequently, debugging is easier, since errors can be localized.

- The use of pipes to support interprocess communication should make moving the system to the VAX/VMS environment relatively easy, since pipes can be replaced by VMS mailboxes.

- A design using communicating processes leads to an implementation that can provide some degree of parallelism in an appropriate hardware environment.

The major drawback to building the system as a collection of communicating processes is that the passing of messages between tasks incurs a certain amount of system overhead. Because the software described in this report is a prototype, this overhead seems a reasonable price to pay. For an operational system, the tradeoffs are less obvious.

3 Software System Description

The major software components of MSPES are the inference system (ERS), the user interface, the graphics package, the geographic information system (QUILT), and the input conversion package. This section discusses each of these components.

3.1 Inference System (ERS)

The inference system within MSPES is ERS, Embedded Rule-based System, which is a system developed by PGSC. ERS uses a goal-directed backward chaining mechanism. It uses probabilistic reasoning, and permits degrees of belief to be propagated using either logical or Bayesian rules. A detailed description of ERS can be found in the ERS User Manual [Barth and Quinn-Jacobs, 1986].

3.1.1 Primitive Management Routines

From the viewpoint of ERS, the interface to the application program consists of calls from ERS to three special primitive management routines. These routines, which are written as part of the application, are as follows:

- **Priminit**: This routine reads the node coordinate input and saves it. On its first call, it starts the primitive processes.

- **Evalprim:** This routine is told by ERS what primitive needs to be called, and sends messages to the appropriate primitive. It then returns the responses sent back by the primitive to ERS for rulebase evaluation.

- **Primclose**—When the last node has been evaluated, this routine causes all primitive processes to shut down.

3.1.2 Rules

The rules used in the prototype MSPES have as their basis the following notion: mines are most likely to be placed in narrow corridors of high mobility through areas of lower mobility. These corridors may be roads, or they may be *canalized* portions of regions. (A region is canalized if it contains a narrow pass, or canal, as illustrated in Figure 3.1.)

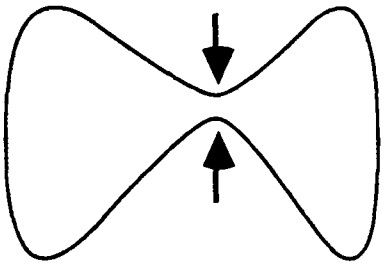


Figure 3.1 A Canalized Region. The Canalized portion is indicated by the arrows.

With some oversimplification, the rules can be summarized as follows. Note that the last three categories—possible, likely, and very likely—are not mutually exclusive, for reasons that are addressed in Section 3.1.4, below.

- A node is classified as *unevaluated* if movement is not possible. (Built-up areas are treated as areas in which movement is not possible.)

- A node is classified as *not likely* if it is in an area in which movement is possible but is over a certain threshold size. In large open areas, minefields are not cost effective.

- A node is classified as *possible* if it is neither *unevaluated* nor *not likely*.

- A node is classified as likely if one of the following conditions is met:

    - movement is possible and the region of mobility is canalized, or

    - a road is present.

- A node is classified as *very likely* if movement is possible, the region of mobility is canalized, and a road is present.

The complete text of the rule base description is presented in Appendix C.

3.1.3 Action Routines

These routines are called by ERS when an evaluation has been reached. They send a message about the evaluation to the standard output.

There is a subtle point about the node evaluation strategy that was alluded to in the previous subsection. A node can be evaluated as being in one of several prediction categories: unevaluated, not possible, possible, likely, or very likely. In the prototype MSPES, the last three categories—possible, likely, or very likely—are not considered to be mutually exclusive. Thus one rule may cause a node to be identified as possible and another to identify it as likely. If this happens, the action routines for "possible" nodes and for "likely" nodes are both invoked, and the messages "Mine possible" and "Mine likely" are both written. The job of interpreting these messages and classifying the node as likely (since "likely" is a more restrictive category than "possible") is performed by the user interface task, MSPTOOL(see Section 2.1).

The reason for the approach just described is that the "likely" nodes are a subset of the "possible" nodes, so each "likely" node should also be identified as "possible." One could argue that a human expert would first identify the "possible" areas, then restrict attention to these areas while identifying the "likely" areas. Attention would in turn be restricted to the "likely" areas when the expert was looking for "very likely" areas.

3.2 GIS (QUILT)

The geographic information system used in MSPES is the QUILT system, developed by the Center for Automation Research at the University of Maryland[Shaffer *et al*, 1987]. QUILT is based on quadtrees [Samet, 1984], and supports the storage and retrieval of region, linear, or point data. QUILT consists of a kernel, which provides a library of fundamental routines required for traversing and manipulating quadtrees, and a collection of programs that access these routines.

Within MSPES, there are eight basic primitives that access the quadtree structure. (These do not include the functions for converting from SLF to QUILT format discussed in Section 3.5). The following four primitives are used by the Inference System as part of the process of creating a manuscript.

- **node_size:** This primitive returns the size (area) in pixels of the quadtree node containing a given point, specified as an (x,y) coordinate pair.

- **ccm_class:** This primitive returns the ccm_class of the quadtree node containing a given point, specified as an (x,y) coordinate pair.

- **canal1:** This primitive returns the canalization class of the quadtree node containing a given point, specified as an (x,y) coordinate pair. A node is canalized if it represents a canalized portion of a region in which mobility is possible. The precise definition used is as follows: a node is canalized in the east-west direction if

  (1) it has at least one neighbor to the east and at least one neighbor to the west in which mobility is possible, and

  (2) it has no neighbors to either the north or the south in which mobility is possible.

  The definition of a node canalized in the north-south direction is analogous. A node is canalized if it is canalized in either the east-west or north-south direction. An example of a node canalized in the east-west direction is shown in Figure 3.2. The implementation of the **canal1** primitive is discussed in Appendix B.

- **road_distp:** This primitive determines whether there is a road within a given threshold distance of a given point, specified as an (x,y) coordinate pair. Its implementation is discussed in Appendix B.

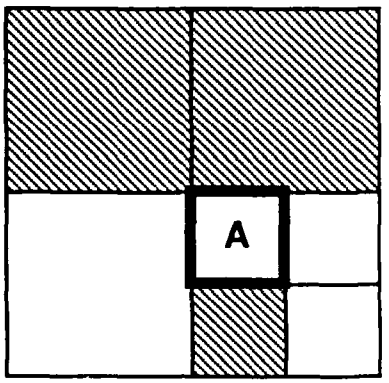


Figure 3.2 A Quadtree Node that is Canalized in the East-West direction.

The following four primitives are used by portions of MSPES other than the **create manuscript** function:

- **node_corner**: This process provides the coordinates of the northeast corner, the extent, and the value (CCM-class) of a node containing a given point. (The point is expressed as an (x,y) coordinate pair.)

- **qdisplay**: This process drives the display of information from areal, lineal (PM), and point quadtrees. It sends a message to the Graphic Interface to display a particular quadtree node.

- **qdmpupd**: This routine dumps node coordinates, gets update records for each node, and posts the updates to a manuscript quadtree. Its interaction with the other processes in MSPES is described in Section 2.1, above.

- **qupdate**: This process reads quadtree node update requests, consisting of node coordinates and update values, from its standard input. It then posts updates to the quadtree as requested.

3.3 User Interface

The User Interface uses SunView, an object-oriented graphics package provided by Sun. The User Interface comprises the following functions.

- **Input Map**: This function controls data conversion.
- **View Map**: This function lets the user look at overlays and manuscripts.
- **Edit Map**: This function lets the user make simple modifications to quadtrees.
- **Create Manuscript**: This function drives the creation of a minefield site prediction manuscript from CCM and LOC overlays.
- **Explain Manuscript**: This function works in a more interactive mode than **Create Manuscript**, and permits user to see ERS results for specific locations of the manuscript.
- **Graphic Interface**: This interface is used by other user interface functions. It controls the map and legend display areas.
- **Explanation Interface**: This interface is also used by other user interface processes. It is the vehicle for communications between the user interface and ERS.
- **Edit Rulebase**: This process provides a text editor interface that lets users make changes in rulebases.
- **Help**: This process provides an on-line help facility. It lets users scroll through help files.

3.4 Graphics Package

MSPES uses SunView and PixRect calls. Both of these packages are distributed by Sun. PixRect provides basic graphics services. SunView provides an additional layer of support, and handles certain window-related services, such as clipping.

The graphics functions used by the User Interface are provided by SunView. The graphics functions required by the map viewing and legend area display functions are provided by a combination of SunView and PixRect routines For map display, the areas of each quadtree leaf node are filled with the appropriate color. For legend area display, text and boxes filled with the color appropriate for the current overlay type are displayed.

3.5 Input Conversion Package

The conversion process from SLF format to QUILT format is illustrated in Figure 3.3. SLF data (which is in ASCII) is converted to a sequence of alternating feature headers and coordinates by the **readslf** program. For areal quadtrees this output is scanned by one of a number of **xxxcolors** programs that determine the feature header order and create a lookup table for the **fillpoly** program to use so that the quadtree will have the right values for nodes. The **readslf** output is converted to a raster (in the case of areal quadtree data) by the program **fillpoly**. Another program, **fillpicholes**, takes this raster and tries to eliminate the holes between polygons Finally, the program **build**, which is a slightly modified version of the QUILT program of the same name, takes this raster and creates a quadtree from it.

In the case of LOC data, the conversion from the **readslf** output to a PM-quadtree is done directly by the program **pmbuild**. Like **build**, **pmbuild** is a slightly modified version of the QUILT program of the same name.

A more detailed description of the conversion process is presented in Appendix A.

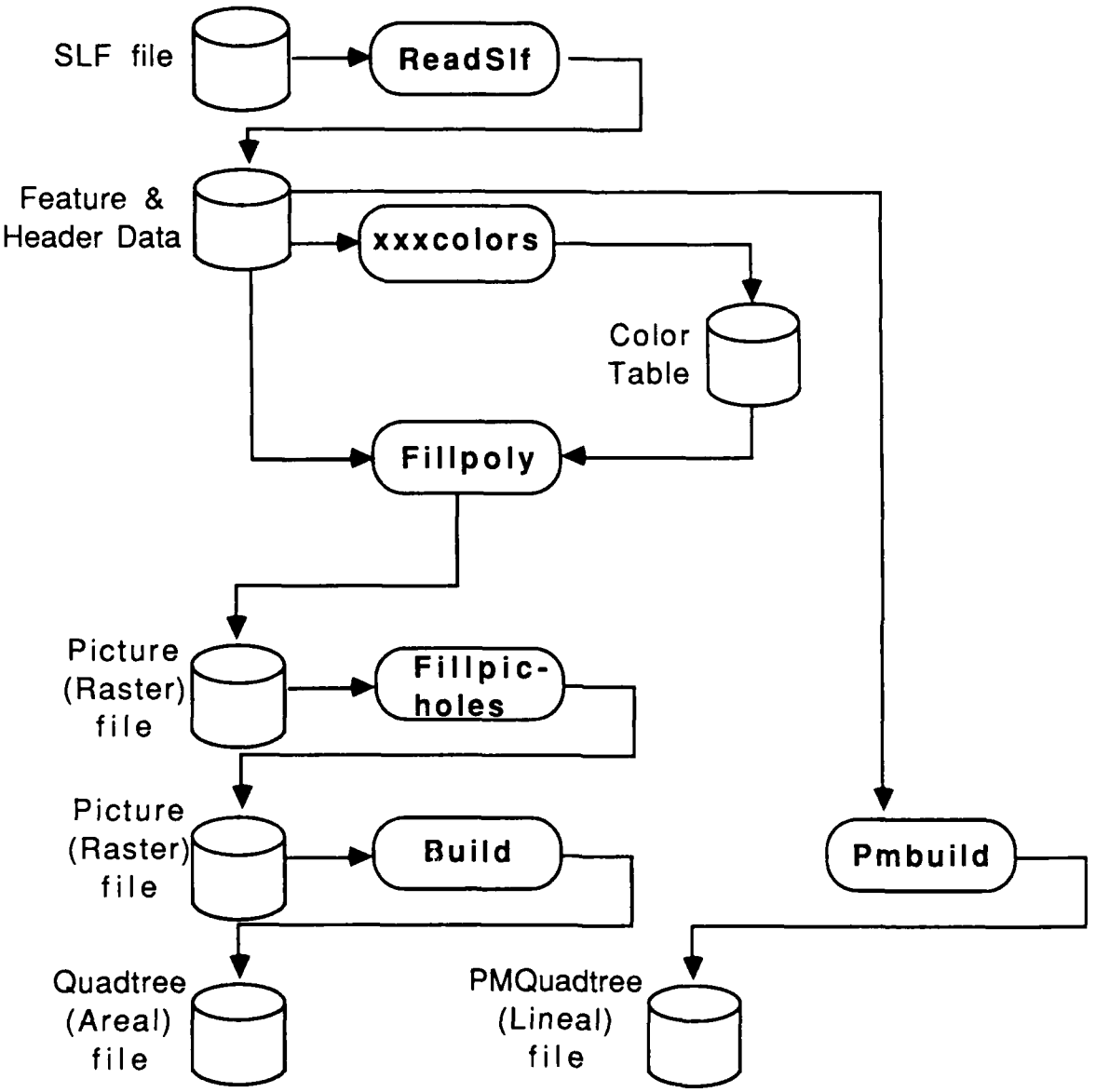


Figure 3.3 The Conversion Process from SLF to Quadtree Format.

4 Evaluation and Recommendations

Based on experience building and exercising the prototype MSPES system, recommendations for enhancements and improvements are proposed. The recommendations are divided into three major categories: recommendations for the overall system architecture, recommendations for the QUILT software, and recommendations for ERS.

Each suggested enhancement is classified as being either a *functional* enhancement or a *performance* enhancement. A functional enhancement addresses system functionality: that is, if incorporated, it would add new capabilities to the system. A performance enhancement addresses system performance: if implemented, it would improve the system by increasing its speed and/or lowering its use of system resources such as memory, disk space, etc.

4.1 Recommendations for System Architecture

To improve performance, it is important that interprocess communication overhead be reduced. The prototype was implemented as a collection of cooperating processes, for reasons discussed in Section 2–2, above. A software architecture of this types necessarily has some input/output (i/o) overhead associated with it. This problem can be addressed in two ways.

- **Combine processes.** Currently each quadtree primitive function is implemented as a separate process to allow the necessary number of quadtree files to remain open during execution of the rule base. Grouping functions together, to reduce the number of pipes, would eliminate some of the interprocess communication overhead. One possible approach is to group everything together in a single process. A less radical alternative would be to group together all quadtree primitives that access the same set of files.

- **Reduce i/o volume.** Currently, the ERS action routines generate action messages, which are written to a pipe. Other information is written to the same pipe by other ERS routines. The routines in MSPTOOL read this output produced by ERS, filter out the "noise," and parse the action messages. This arrangement incurs unnecessary overhead. The overhead can be reduced

considerably by having a separate pipe used strictly for action messages, and by streamlining the messages.

Portability can be improved by converting the graphics and user interface functions from SunView to X Windows. The X Window System [Scheifler and Gettys, 1986] was developed at MIT, and supports machine independent, device independent, high-level, window-oriented graphics. Increased portability will permit MSPES to be ported, with minimal effort, to virtually all hardware environments. In particular, conversion of the graphics portion to X Windows should speed up the process of moving MSPES from the Sun to the VAX GPX Workstation environment.

From a functional point of view, the major improvement that can be made to the system is to make the geographic reasoning capability more sophisticated. One very easy modification is making the threshold distance for the **road_distp** a rule parameter, rather than a constant. A more difficult, and critical, enhancement is improving the canalization concept. This concept has several shortcomings. In the prototype MSPES, no distinction is made between a pass into a "blind alley" (which would be an unlikely mine site) and a pass between two large portions of the region (which would be a likely mine site.) To differentiate between these two situations, some global knowledge about the shape of the region is required.

One promising approach to describing the shape of a region is to use the skeleton, or medial axis transform (MAT), as a measure of shape. The MAT of a region, originally introduced in [Blum, 1967], consists of a set of curves with the property that for each point on the curve, the distance from the point to the boundary of the region, in a direction normal to the curve, is a local maximum. More precise definitions and efficient algorithms for computing the skeleton can be found in [Zhang and Suen, 1984] and [Lee, 1983]. Efficient algorithms for computing MAT-like transforms for quadtrees have also been developed[Samet, 1983]. By computing the MAT, and storing certain additional information (e.g. the distance to the boundary in the directions normal and tangent to the direction of the MAT curve), rules can be devised to detect and exploit a more robust notion of canalization.

4.2 Recommendations for QUILT

Within QUILT, the following functional and performance enhancements are recommended:

- **Saving LOC feature header information.** Currently, QUILT makes no provision for attribute information to be associated with segments in the PM quadtree. As a result, the prototype MSPES cannot distinguish between the various types of LOC features (e.g. roads vs. powerlines). For the prototype, this problem was ignored, and all LOC features were treated as roads; clearly, this is not an acceptable approach for the operational system. There are two possible ways of addressing this problem: enhancing QUILT, or building a parallel segment table with additional feature information. Because the second approach is simpler, it is the approach most likely to be taken.

- **Using LOC feature header information.** The QUILT primitives must be updated to use feature information derived from the LOC feature header. For instance, the primitive **road_distp** should only look at roads, not power lines. This enhancement is closely connected with the preceding one.

- **Use of real-world distances.** In the prototype MSPES, all computations are done using distances measured in pixels, rather than in "real-world" units (e.g. meters). This means that the rules must be formulated in terms of pixel sizes, which is an unreasonable constraint to place on the expert and/or the knowledge engineer formulating the rules. It also means that the rules are tied to a fixed quadtree leaf size. If the quadtree primitives can accept and return distances measured in real-world units, then the rules can be formulated in real-world units.

- **Speed up the road_distp primitive.** Although the quadtree primitives are not the system bottleneck within the prototype MSPES, there is room for improvement in their performance, particularly for the **road_distp** primitive. In Appendix B, the "top-down" algorithm that was used to implement this primitive is described and a "bottom-up" alternative is sketched. The bottom-up version, while somewhat more complicated to code, might provide better performance.

4.3 Recommendations for ERS

ERS appears to be the primary system bottleneck within the prototype MSPES. It appears that the speed of ERS could be increased considerably by incorporating the following performance improvements:

- **Simplify the Evidence Selection Mechanism.** Execution profile information on both the SUN and VAX versions of ERS shows that much of the processing time is spent in routines that implement the evidence selection mechanism. This mechanism is currently designed to search the entire inference net for the evidence that could potentially have the greatest effect on the degree of belief of the current goal. A simpler mechanism that uses local criteria to drive a heuristic search would probably run much faster and be just as effective. The evidence selection mechanism should be designed to allow the knowledge engineer or user to specify the amount and detail of considerations taken into account during evidence selection.

- **Eliminate or reduce floating point operations.** Profiling information indicates that a significant amount of time is spent in routines that calculate the degrees of belief for Bayesian nodes, and in routines that perform conversions between probabilities, odds, and degrees of belief. These operations all make heavy use of floating point arithmetic. ERS could be modified to use only integer arithmetic by using conversion tables and changing the piecewise linear interpolation functions to be step functions. Changing the degree of belief measure from being a continuous function to one with, perhaps, two or three hundred discrete values, would satisfy the requirements for MSPES and many other applications, in terms of the necessary resolution of the uncertainty measure.

- **Overall code improvements.** There are several areas in which the ERS code could be made more efficient. In particular, the organization of the inference net, and the manner in which it is accessed, could be improved. The command functions implemented to support the MSPES explanation interface could be replaced by a set of functions to query the inference net.

Appendix A: Conversion of SLF Data to QUILT Format

This appendix describes the software available to convert Standard Linear Format (SLF) data to QUILT format, and to examine and manipulate the data at various stages of the process. The programs described here are not all part of the conversion process outline in Section 3–5. For example, the program **chkslf** lists the contents of an SLF file, but does not do any format conversion. Some of the software was written by PGSC, some was part of the QUILT system, and some QUILT routines have been modified by PGSC. An introduction to SLF and a description of the data sets provided for Task 2 of Phase 1 are included as background material.

A.1 Standard Linear Format Data

Overlays for the test area were supplied in the Defense Mapping Agency's (DMA) Standard Linear Format (SLF). SLF was designed as an exchange format for DMA's planimetric and hypsometric feature data. The SLF structure is a topologically structured vector format which contains records for point, line and areal features. The SLF data structure is composed of a Data Set Identification (DSI) record, Segment (SEG) records, Feature (FEA) records and a trailing Text (TXT) record. The DSI record contains information on the content of the SLF data set. The SEG records contain segment orientation and coordinate data. The FEA records contain a description of point, line and/or area features. The TXT record is a terminator for the SLF data set and contains free-form text commentary.

A.2 Overlay Data Sets

The following overlays were supplied in SLF format for input into the MINES system:

- Surface Configuration (SLOPE)
- Transportation, or Lines of Communication (LOC) for the same coverage areas as the SLOPE overlays.
- Vegetation overlays. The coverage areas for the vegetation overlays were different from the coverage areas for the SLOPE and LOC overlays.

The SLOPE data consists of areal features delineating surface configuration data. Features were categorized into classes by percentage of slope. Classes are 0-3%, 10-20%, 20-30%, 30-45% and greater than 45%.

The vegetation overlay consists of area features delineating the possible classes of coniferous/evergreen forests, deciduous forests, mixed (coniferous/deciduous) forests, orchard/plantation, forest clearing, swamp, march/bog, wetlands, vineyards/hops, bamboo, bare ground, agriculture dry crops, agriculture wet crops, agriculture terraced crops (wet and dry), agriculture shifting cultivation, brushland/scrub (less than 50% closure), brushland/scrub (greater than 50% closure), grassland (pasture, meadow), grassland (with scattered trees and scrub), intermittent open water, and perennial open water.

The transportation overlay consists of the highway features: road, bridge, tunnel, ferry, on-route ford and overhead drop; the rail features: railroad, bridge, tunnel, ferry and overhead drop; and the airfield features: runway and landing area. The overlay in the test area consisted of linear features only.

A.3 Conversion of Overlay Data to Quadtrees

The overlay data in SLF form is converted to quadtree format by a series of programs. The vector features are first extracted from the SLF data set. Features are coded by a user selected criteria and converted into a raster file, with area features filled and linear features drawn. The raster data is then converted into quadtree format. In addition to the basic conversion programs, illustrated in Figure 3–1 of this report, there also exist verification, filtering and subscene extraction utilities to assist the conversion process.

The **chkslf** module is used to produce a report of the contents of an SLF file. The report may list the DSI record, segment records, and/or feature records.

The **readslf** module creates segment and feature data from SLF data. It originally was developed by University of Maryland (UOM). The software was modified as follows:

- The software takes into account that elevation data (encoded as 0) may be present.
- The software handles segment numbers that are skipped or missing.

- Since the border of the data set is not always explicitly defined, area features with border segments are explicitly closed if necessary.

- Enhancements were made to allow for scaling of the output data to a ground sample distance or a fixed image size. The module also allows for the entering of a scale factor number.

- The option to create a feature attribute file from the SLF data for the extracted feature data is provided.

If a feature attribute file is created it may be viewed with the **dsphdr** program. The program lists feature numbers and the attribute feature codes to the standard output.

Extracted SLF area features are converted to a picture (raster) format with an enhanced version of the UOM **fillpoly** program. The module fills area features with a numeric code. The code assigned to a feature is determined by input selection criteria. The selection criteria may use feature number, extracted attribute data, a user supplied color table, or a default color table. PGSC enhancements to the program allow for automatic sizing of the output raster to the actual size of the scaled segment data, and for the coding of features to attributes that were previously created by **readslf**.

The **fillpoly** program can leave gaps in the data between features. The gaps consist of single pixels or small clumps of isolated pixels. Module **fillpicholes** may be run on the raster file to fill in the gaps that were not coded. The program applies a median filter over a 3 by 3 window to determine codes of raster points not coded.

Extracted SLF linear features are written to a picture file with the **filline** program. Raster points are computed from endpoints of vectors within a segment. Raster coding and scaling is performed identically to the way **fillpoly** operates.

The module **ramrast** strips the UOM CVL header from a picture file and creates a raw 8-bit raster file. The data created may then be used in off-line applications that do not need the CVL header.

Subscene or superscene extraction of the picture file is performed from the **picthis** module. The current geographic extent of the file is listed and southwest and northeast corners may be entered to define a new picture file.

The UOM **build** module converts a picture file to a quadtree format. The name of the picture file for input and the name of the output quadtree are arguments. An option is provided to have the module reports the current line being processed to the screen. (This is a PGSC modification; the original UOM module always provided this report.)

Appendix B: Enhancements to QUILT

For the MSPES prototype, several primitives have been written that read and/or manipulate quadtree files, using routines from the QUILT library, and communicate via interprocess i/o with other processes in the system. These primitives and their functions were listed in Section 3. For the convenience of the reader, they are repeated here:

- **node_size** returns the size (area) of the quadtree node containing a given point, specified as an (x,y) coordinate pair.

- **ccm_class** returns the ccm_class of the quadtree node containing a given point, specified as an (x,y) coordinate pair.

- **canal1** returns the canalization class of the quadtree node containing a given point, specified as an (x,y) coordinate pair.

- **road_distp** determines whether there is a road within a given threshold distance of a given point, specified as an (x,y) coordinate pair.

- **node_corner** provides the coordinates of the northeast corner, the extent, and the value (CCM-class) of a node containing a given point. [The point is expressed as an (x,y) coordinate pair.]

- **qdisplay** drives the display of information from areal, lineal (PM), and point quadtrees.

- **qdmpupd** dumps node coordinates, gets update records for each node, and posts the updates to a manuscript quadtree.

- **qupdate** reads quadtree node update requests, from its standard input, and posts updates to the quadtree as requested.

Many of these routines—**node_size**, **ccm_class**, **node_corner**, **qdisplay**, **qdmpupd**, and **qupdate**—are straightforward adaptations of QUILT routines, and their implementations are not discussed here. The remaining two routines—**canal1** and

road_distp—increase the functionality of QUILT, and their implementations are discussed below.

The MSPES primitives use a file that is not part of QUILT, called a quadtree header file. Many of the above primitives open the the header file to determine the name of the (QUILT) quadtree file to be opened.The quadtree header file is used primarily as a means of keeping track of the various related quadtrees that support a manuscript. It could also be used as a place to store additional information about the quadtrees that is not directly representable in QUILT, such as the dimensions of a pixel in real-world units.

B.1 The CANAL1 primitive

The **canal1** primitive determines the canalization class of the quadtree node containing a given point, specified as an (x,y) coordinate pair. A node is canalized if it represents a canalized portion of a region in which mobility is possible. More precisely, a node is canalized in the east-west direction if

(1) it has at least one neighbor to the east and at least one neighbor to the west in which mobility is possible, and

(2) it has no neighbors to either the north or the south in which mobility is possible.

North-south canalization is defined analogously, with "north" and "south" replacing "east" and "west" (and conversely). **Canal1** returns one of three values, depending on whether a node is not canalized, east-west canalized, or north-south canalized. (Notice that it is impossible for a node to be simultaneously north-south canalized and east-west canalized.)

Canal1 traverses the four borders of the quadtree node, looking at all immediate neighbors, to determine which neighboring cells support mobility. This traversal provides it with sufficient information to determine whether conditions (1) and (2), or their analogues for north-south canalization, are satisfied.

The border traversal uses the QUILT library function **QKrneigh** for finding neighbors in a particular corner. **QKrneigh** takes four arguments—a node address, a side direction, a corner modifier, and a maximum node depth—and returns a neighbor node that is

adjacent to the given node in the neighbor side. If the node has several neighbors in that direction, the neighbor node returned is the one on the indicated corner. For example, in Figure B.1, the neighbor of node A on the north side, west corner is node B. For the purposes of the **canall** primitive, the maximum node depth is always set to the maximum depth of the quadtree, which means that a leaf node will always be returned.

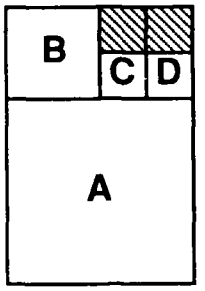


Figure B.1 A quadtree node with several neighbors.

The border traversal along the north border proceeds as follows. (Traversals along the other three borders are similar; the code is actually driven by a four-entry table with one entry for each border.) The first neighbor node is the neighbor on the north side, west corner (node B in Figure B.1). Once the first neighbor is found, the next neighbor node is the neighbor of the current neighbor node on the east side, south corner. Processing stops when the eastern edge of the current neighbor node is at least as far east as the eastern edge of the original node. In Figure B.1, this processing causes a total of three neighbor nodes to be visited, namely B, C, and D. These nodes are exactly the nodes adjacent to node A along the north border.

The situation illustrated in Figure B.1 is in some sense the worst case, since node A has several neighbors. In the two situations illustrated in Figure B.2, the node node A has one neighbor to the north. Because of the way the rules work, **canall** is generally called for small nodes, so one might expect these two situations to be more common.

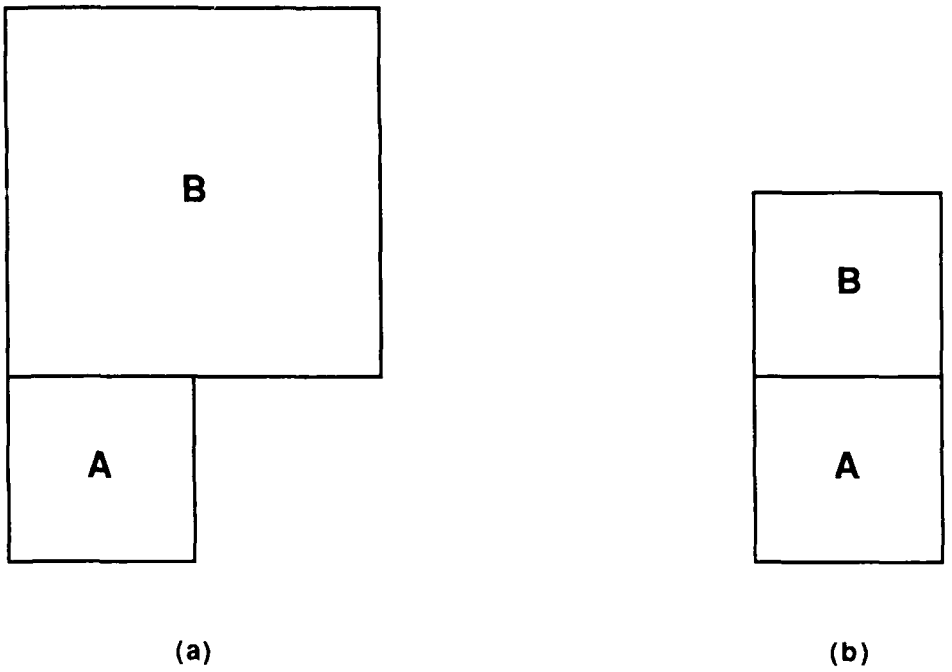


Figure B.2 Quadtree nodes with only one neighbor

B.2 The ROAD_DISTP primitive

The **road_distp** primitive determines whether there is a road within a threshold distance of a point. The implementation used in the prototype MSPES is based on a guided, top-down search. An implementation based on a bottom-up search, using neighbor-finding, is also possible: such an algorithm is sketched at the end of this section.

The idea behind the algorithm is to start at the root, recursively search all descendant nodes in increasing distance from the given point, and to stop when either a road within the threshold distance is found or the tree is exhausted. The search is pruned so that only those portions of the tree that could contain a road within the threshold distance are examined. The algorithm makes use of a stack that contains, at any time, only nodes that have not been visited and that could conceivably contain a portion of a segment within the threshold distance.

The algorithm is best illustrated by means of an example. In Figure B.3, the quadtree representing the illustrated 32 by 32 area is to be searched to determine whether there is a road within seven units of the point (21,22). The figure shows the threshold circle and the five segments that constitute the road network.

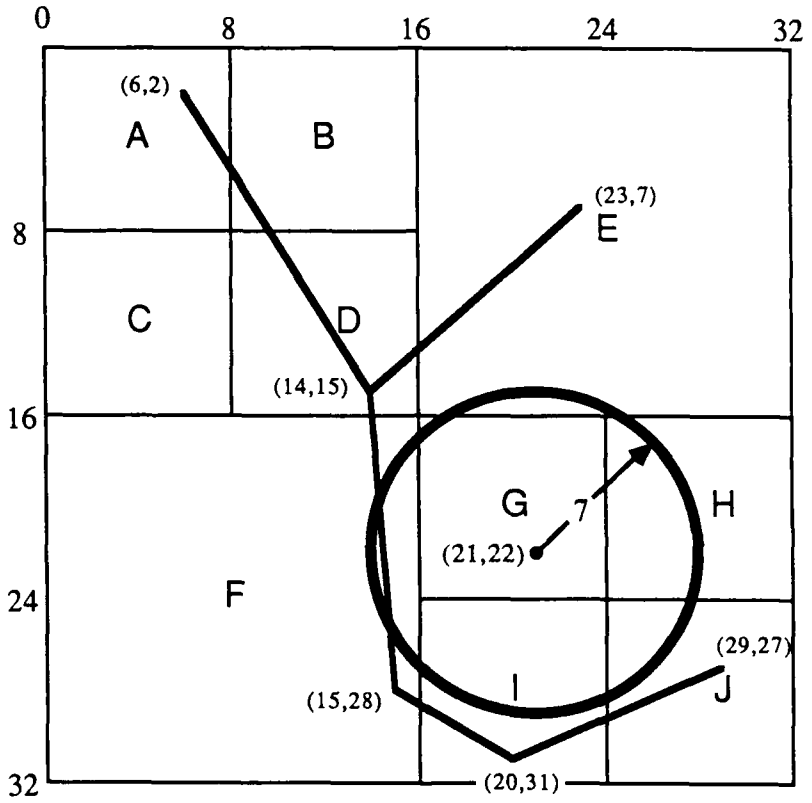


Figure B.3 Example of the algorithm for the primitive road_distp.

The algorithm first examines the root node and determines its sons are to be visited in this order: southeast, southwest, northeast, northwest. (This is done by arranging the nodes according to the smallest distance from the target point to a point in the node: the closest distances from the point (21,22) to any point in the node is 0 for the southeast son of the root, 5 for the southwest son, 6 for the northeast son, and approximately 7.8 for the northwest son.) So the southeast son is to be visited next. Since the closest distance from

the northwest son to the target point exceeds the threshold distance, the northwest son is not placed on the stack of nodes to be visited. (In effect, the tree is pruned so that none of its descendants will be visited.) The northeast and southwest sons are pushed on the stack of nodes to be visited. The southwest son is pushed second, so it will be popped first.

The southeast son of the root is now visited, and it is determined that its sons should be visited in the order **G**, **I**, **H**, **J**. So **I**, **H**, and **J** are stacked, and **G** is noted as the node to be visited next.

G is a leaf node, so the set of segments running through it are examined. Since there are none, the next node on the stack, **I**, is visited. Node **I** contains two segments, but neither of these are within the threshold distance of the target point. **H** and **J** are visited next, neither yielding a suitable segment.

At this point, the stack consists of the two sons of the root that were placed there in the first step. The next node popped is the southeast son of the root, node **F**. This node contains portions of two segments, one of which intersects the threshold circle, so the search is successful. A total of 5 leaf nodes—**G**, **I**, **H**, **J**, and **F**—have been visited. Node **E** was not visited because the search terminated before it got there, and the remaining nodes were not visited because of the tree pruning that occurred.

In the actual coding of the algorithm, a minor modification to the obvious implementation has been observed to significantly improve the run time. Instead of computing the distance from the target point to a segment and comparing it with the threshold distance, the squares of the quantities are compared. This means that square roots do not have to be computed; moreover, all arithmetic can be done using integers. (On a successful query, the distance to the first segment found within the threshold distance, which is not necessarily the distance to the closest segment, is returned. This requires computing an integer square root. Since it seemed wasteful to link in the floating point library for this one computation, a special routine was written. This routine computes an integer square root by making several comparisons to find an appropriate starting point and then either doing a table lookup, if the argument is small, or using Newton's method. Things are arranged so that at most two iterations of Newton's methods are necessary for all integers less than 10,000.)

An alternative approach to finding segments within a threshold distance could be based on neighbor-finding. For example, after determining that the target point was in node **G**, the algorithm would search **G**, and then its neighbors. The advantage to this approach is that the overhead of starting at the root is avoided; searching starts at the node containing the target point. The disadvantage is that an algorithm based on this approach appears to be more complicated that the top-down approach. The complication arises if an appropriate segment is not in the node containing the target point or any of its neighbors: it becomes less obvious how to store, and search, the remaining nodes. However, if this problem can be addressed, an algorithm based on neighbor-finding would probably be faster than the one used in the prototype.

Appendix C: The Rulebase for the Prototype MINES System

This appendix contains the source listing of the rulebase for the prototype MINES system. An explanation of the ERS rule description syntax can be found in the *ERS User Manual* [Barth and Quinn-Jacobs, 1986].

```
RuleBase    Minefields

Version     Prim_choice

;
; J. Doughty, 13 Jan 1988
; This version of the minefield site prediction rulebase has been
; modified so that all rules use unitless measures or are with respect
; to QUILT pixel distances.
;
; Based on concept demo rulebase by S. Barth, 28 April 1987
; which was based on an initial rule base built by O. Long, November '86
; and on a flow chart by A. Downs, April '87
;
; The purpose of this rule base is to determine the likelihood of a
; minefield being present at a certain site. Four categories of
; likelihood are assigned, very likely, likely, possible, and unevaluated,
; each represented by a separate goal node. Unevaluated means that
; not enough evidence is present to assign one of the other categories.
; (For example, regions that are built-up areas are unevaluated)
; The rules operate in the framework of a current region having been
; selected from the quadtree, for which a likelihood assignment will be made.
; The only basic evidence used is LOC and CCM data, plus spatial
; relations about quadtree regions (e.g. adjacency)
;
; The actions that are fired when a goal is reached will update a
; minefield site prediction manuscript with the goal value for that area.

ActionSet    GoalActions   10.0 Any

InitialGoal     unevaluated

; The goals "unevaluated" and "not_likely" have simple criteria, and
; the other 3 goals, "possible", "likely", and "very_likely" will be
; considered only when these criteria are not met.
; "Possible", "likely" and "very_likely" are treated as sub-categories
; of likelihood; i.e., every very likely site is also a likely site, and
; every likely site is also a possible site. Therefore, for whatever
; evidence is present, db(very_likely) <= db(likely) <= db(possible),
; (db stands for degree of belief).

;
; Eventually, goal priors should be set by the area being considered,
; since certain types of terrain may predominate. And if priors are
; interpreted as the expectation of areal coverage by each likelihood
; category, then very_likely should have the lowest prior expectation, since
; minefields won't cover most of the area.
```

```
;

node unevaluated
    member    GoalActions
    action    update_UNEVAL
    text desc
        " more data is needed to evaluate the current region "
    elaboration
        "  a region is unevaluated if its CCM class indicates movement is not
           possible or if it is a built up area"
    explanation
          "   a region is not evaluated as to the likelihood its being a
           minefield site if its CCM class indicates movement is not possible
           or if it is a built up area"
    inference
        prior -5.0
        logical antecedents or ( unev_nogo unev_builtup )
        control
                goal

node possible
    member    GoalActions
    action    update_POSSIBLE
    text desc
        " the current region is a possible minefield site"
    elaboration
          " possible minefield sites are regions where a minefield MIGHT be
           located"
    explanation
        " Nearly everything is a possible minefield site, unless it's
an unevaluated area, but we rule out regions that are too large, since
minefields would be ineffective in large areas where CCM is possible "
    inference
        prior 1.0
        bayesian antecedents (
                ccm_possible    pw 20.0         nw -1.0
                )
        control
                context of unevaluated int min 0.0
                context of not_likely  int min 0.0
                context of rs_lt_limit  int 0.0 max
                goal


; For the likely goal, antecedent weights were assigned with the idea that
; if all three conditions are present we want db 20, if only ccm_possible
; and road_loc are present we'll say db 10, if ccm_possible and canal_area,
; but there's no road, db 10. If only ccm_possible occurs then db 0
; (50% chance)
; The other cases don't occur, since we use ccm_possible as context.
;
node likely
    member    GoalActions
    action    update_LIKELY
    text desc
        " the current region is a likely minefield site"
    elaboration
```

```
        " CCM is possible and movement is canalized or a road is present in
           regions that are likely minefield sites"
    explanation
        " Likely minefield sites are where movement is canalized, and/or
a road is present, and CCM is possible. "
    inference
        prior -5.0
        bayesian antecedents
                (
                ccm_possible    pw 5    nw 0
                canal_area      pw 10   nw 0
                road_loc        pw 10   nw 0
                )
        control
                context of unevaluated int min 0.0
                context of not_likely  int min 0.0
                context of ccm_possible int 0.0 max
                goal

node very_likely
    member    GoalActions
    action    update_VERYLIKELY
    text desc
        " the current region is a very likely minefield site"
    elaboration
         " Regions on roads where movement is canalized are considered very
           likely minefield sites"
    explanation
          " For now, only sites on roads where movement is canalized are
           considered very likely"
    inference
        prior -5.0
        bayesian antecedents
                (
                fast_ccm        pw  5   nw  0
                canal_area      pw 10   nw  -5
                road_loc        pw 10   nw  -5
                )
    control
        context of unevaluated int min 0.0
        context of not_likely  int min 0.0
        context of fast_ccm int 0.0 max
        goal

; Currently the only condition for a region being not_likely is if
; it's too large an area where ccm is possible
;
node not_likely
    member    GoalActions
    action    update_NOTLIKELY
    text desc
        " the current region is not a likely minefield site "
    elaboration
         " A region is not a likely minefield site if CCM is possible but the
           region is so large that minefields would be ineffective."
    explanation
        " It's not likely that minefields would be employed in large open
areas "
```

```
    inference
        prior -10.0
        bayesian antecedents (
                rs_ge_limit     pw 20.0 nw -10.0
                )
        control
                goal
                context of ccm_possible int 0.0 max

;_____________________________________________________________________

; Intermediate Hypotheses
;

; Any category for CCM, except no-go's or built-up is considered as
; possible
;
node ccm_possible
    text desc
        " CCM data indicates that movement is possible in the current region"
    elaboration
         " CCM movement is possible in regions that are classified as 'go',
            'restricted', 'slow', or 'very slow'."
    inference
                prior -10.0
                logical antecedents or ( go restricted slow very_slow )

node unev_nogo
        text desc
        " CCM data indicates that the current region is a no-go area "
        inference
                prior -5.0
                logical antecedents or ( no_go no_go_water )

node unev_builtup
        text desc
        " CCM data indicates that the current region is a no-go built-up area"
        inference
                prior -10.0
                logical antecedents and ( built_up )

; node slow_ccm
;       text desc
;       " CCM data indicates that the current region is Slow (3) or
; Very Slow (4) - possible tank speeds are 1.5 - 15 km/hr."
;       inference
;               prior -10.0
;               logical antecedents or ( slow very_slow )

node fast_ccm
    text desc
        " CCM data indicates that fast CCM is possible"
    elaboration
         " CCM data indicates that the region is Go (1) or Restricted (2) -
            possible tank speeds are above 15 km/hr. "
        inference
                prior -10.0
                logical antecedents or ( go restricted )
```

```
node road_loc
        text desc
        " LOC data indicates that a road is present in the region "
        elaboration
        " LOC data includes roads, railroads, power lines, and other
man-made lines of communication."
        inference
                prior -10.0
                logical antecedents and ( near_road )

node canal_area
        text desc
        " the current region is part of a canalized area "
        elaboration
        " Canalized areas are where CCM movement is constrained by natural or
            man-made obstacles into a channel in one direction or another "
        inference
                prior -10.0
                logical antecedents and ( canal_evid )

;
```

```
; Evidence Nodes
;

choicenode region_size
        text desc
                format specify exclusive
                " the size of the current region (in map units)"
        test node_size
        inference
            prior 0.0
            answers
                >= 60 : ( rs_ge_limit )
                <  60 : ( rs_lt_limit )
        control
                notgoal

node rs_ge_limit
     text desc
        " region size >= 60 map units"
     explanation
      " The size of the region is greater than or equal to 60 map units "
     inference
        prior 0.0
;       test node_size >= 60

node rs_lt_limit
     text desc
        " region size < 60 map units"
     explanation
        " The size of the region is less than 60 map units "
     inference
       prior 0.0
;      test node_size < 60
```

```
; SWB 20 Oct. 1987 - a choice node with primitive, this eliminates
; having to have each node for a set of mutually exclusive function results
; call the same primitive function. With the "test ccm_class" added to
; the choice node, ccm_class is called once, and the resulting value is
; compared to the specified answers.

choicenode ccm
    text desc
        format specify exclusive
        " the CCM category of this area"
    test ccm_class
    inference
        prior 0.0
        answers
             = 1 : ( go )
             = 2 : ( restricted )
             = 3 : ( slow )
             = 4 : ( very_slow )
             = 5 : ( no_go )
             = 6 : ( no_go_water )
             = 7 : ( built_up )
         control
            notgoal

node go
     text desc
        " region is a go area "
     explanation
        " The region can support speeds greater than 30.0 km/hr"
     inference
        prior -10.0
;       test ccm_class = 1

node restricted
     text desc
        " region is restricted "
     explanation
        " The region allows speeds between 15.0 and 30.0 km/hr"
     inference
        prior -10.0
;       test ccm_class = 2

node slow
     text desc
        " region is a slow travel area "
     explanation
        " The region permits speeds between 5.0 and 15.0 km/hr"
     inference
        prior -10.0
;       test ccm_class = 3

node very_slow
     text desc
        " region is a very slow travel area "
     explanation
        " The region permits speeds between 1.5 and 5.0 km/hr"
     inference
        prior -10.0
```

```
;       test ccm_class = 4

node no_go
     text desc
        " region is no go"
     explanation
        " The region permits speeds less than 1.5 km/hr"
     inference
        prior -5.0
;       test ccm_class = 5

node no_go_water
     text desc
        " region is no go - open water"
     explanation
        " The region contains open water that cannot be crossed"
     inference
        prior -10.0
;       test ccm_class = 6

node built_up
     text desc
        " region is built up"
     explanation
        " CCM category for battle tank is 7 - BUILT UP AREA "
     inference
        prior -10.0
;       test ccm_class = 7

node near_road
     text desc
        " region is near to road LOC"
     explanation
        " LOC data indicates a road is within 3 map units "
     inference
        prior -10.0
        test road_dist < 3

node canal_evid
     text desc
        " the region has evidence of being a canalized area"
     explanation
        " CCM data indicates the region has characteristics of a
canalized area "
     inference
        prior -10.0
        test canal_region > 0

stop
```

Appendix D: List of References

Barth, S. W. and Quinn-Jacobs, D. P., 1986. *ERS User Manual*, PAR Government Systems Corporation, New Hartford, NY.

Barth, S., Downs, A., Long, O., and Stock, J., 1987. *Software System Description for Minefield Site Prediction Expert System*, U. S. Army Corps of Engineers, Engineer Topographic Laboratories, Ft. Belvoir, VA, Technical Report ETL-0449.

Blum, H., 1967. A Transformation for Extracting New Descriptors of Shape. In Walthen-Dunn, W., ed., *Models for the Perception of Speech and Visual Form*, MIT Press, Cambridge, MA, pp. 362–380.

Falls, R.A., 1983. *Using Terrain Analysis to Predict Likely Minefield Sites*, U. S. Army Corps of Engineers, Engineer Topographic Laboratories, Ft. Belvoir, VA, Technical Report ETL-0325.

Lee, D.T., 1982. Medial Axis Transformation of a Planar Shape, *IEEE Transactions on Pattern Analysis and Machine Intelligence 4* (4), 363–369.

Samet, H., 1983. A Quadtree Medial Axis Transform, *Communications of the ACM 26* (9), 680–693. (Also, see Corrigendum, *CACM 27* (2), 1984, page 151.)

Samet, H., 1984. The Quadtree and Related Hierarchical Data Structures, *Computing Surveys 16* (2), 187–260.

Shaffer, C. A., Samet, H., and Nelson, R. C., 1987. *QUILT: A Geographic Information System Based on Quadtrees*, Computer Science Department, University of Maryland, College Park, MD, Technical Report CS-TR-1885.

Scheifler, R. W. and Gettys. J., 1986. The X Window System, *ACM Transactions on Graphics 5* (2), 79–109.

Zhang, T.Y. and Suen, C.Y., 1984. A Fast Parallel Algorithm for Thinning Digital Patterns, *Communications of the ACM 27* (3), 236–239.

Appendix E: Terms and Abbreviations

CCM	Cross Country Movement
CVL	Computer Vision Laboratory
DMA	Defense Mapping Agency
DSI	Data Set Identification
ERS	Embedded Rule-Based System
ETL	U. S. Army Engineer and Topographic Laboratories
FEA	Feature
GIS	Geographic Information System
GPX	A trademark of Digital Equipment Corporation (Graphics Board)
i/o	input/output
LOC	Lines of Communication
MAT	Medial Axis Transform
MIT	Massachusetts Institute of Technology
MOSS	Map Overlay and Statistical System
MSPES	Minefield Site Prediction Expert System
MSPTOOL	The component of MSPES that provides the top-level logic that drives the system, the user interface, and the graphics interface
PGSC	PAR Government Systems Corporation
PM	Polygonal Map (A type of quadtree used to store linear data)
QUILT	A GIS developed by the University of Maryland Center for Automation Research
SEG	Segment
SLF	Standard Linear Format
TXT	Text
UNIX	A trademark of Bell Laboratories
UOM	University of Maryland
VAX	A trademark of Digital Equipment Corporation

END

DATE

FILMED

DTIC

JULY 88